SPSS Graphics

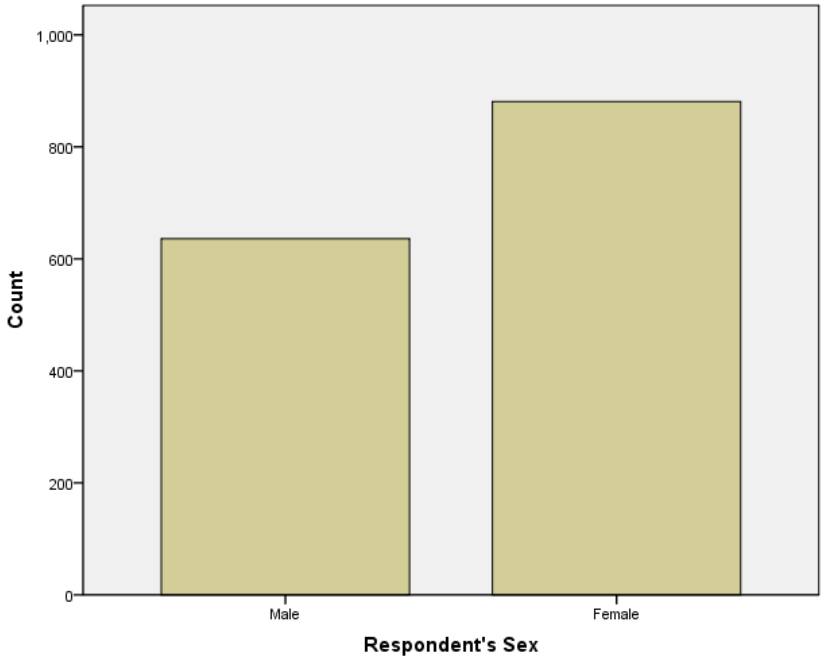
Two Graphics Systems

- Graph
- Ggraph & GPL

- (deprecated) Igraph

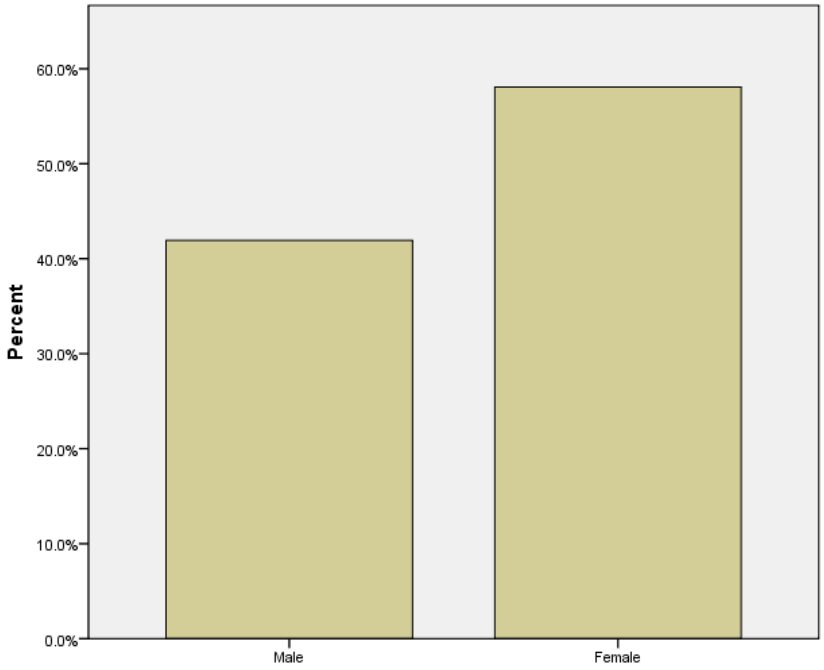
Graph Bar

- frequencies variables=sex.
- graph /bar = sex.



Ggraph Bar

```
GGRAPH
  /GRAPHDATASET NAME="graphdataset"
       VARIABLES=sex COUNT()
  /GRAPHSPEC SOURCE=INLINE.

BEGIN GPL
  SOURCE: s=userSource(id("graphdataset"))
  DATA: sex=col(source(s), name("sex"), unit.category())
  DATA: COUNT=col(source(s), name("COUNT"))
  GUIDE: axis(dim(2), label("Percent"))
  ELEMENT:
       interval(position(summary.percent(sex*COUNT)))
END GPL.
```



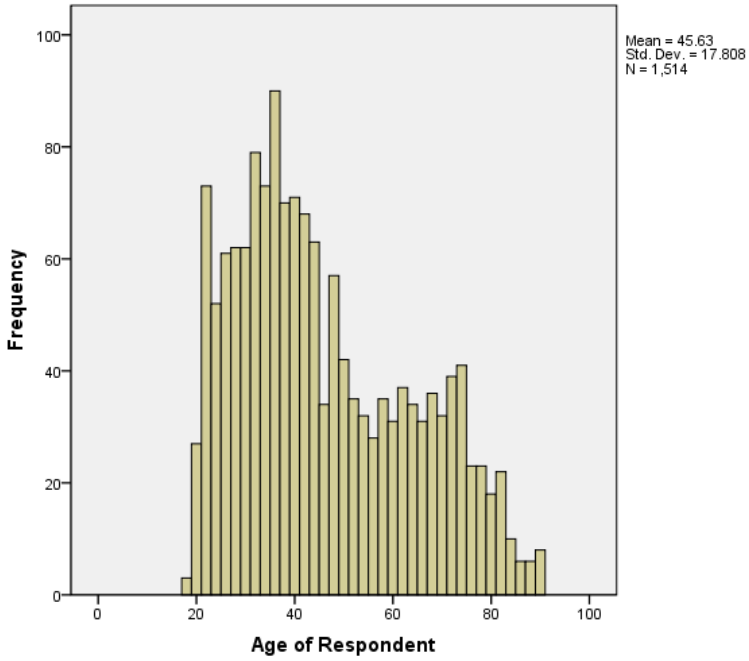
Graph vs Ggraph

- Graph
  - Simple syntax
  - Layering elements is often not possible
  - Aesthetics are controlled by using the Chart Editor and/or templates
- Ggraph
  - More complicated syntax
  - Graphical elements can be layered
  - Aesthetics are controlled by GPL syntax

Graph Histogram

- graph /histogram = age.

- You can add a normal curve
- You can only control binning through the Chart Editor



Ggraph Histogram

```
GGRAPH
  /GRAPHDATASET NAME="graphdataset" VARIABLES=age
  /GRAPHSPEC SOURCE=INLINE.

BEGIN GPL
  SOURCE: s=userSource(id("graphdataset"))
  DATA: age=col(source(s), name("age"))
  GUIDE: axis(dim(1), label("Age of Respondent"))
  GUIDE: axis(dim(2), label("Frequency"))
  ELEMENT: interval(position(summary.count(bin.rect(age,
        binWidth(5)))))
  ELEMENT: line(position(density.normal(age)),
        color(color.blue))
  ELEMENT: line(position(density.kernel.epanechnikov(age)),
        color(color.red))
END GPL.
```

Ggraph Histogram (cont.)

- Direct control of binning
- Layer different theoretical curves, or kernel density curves
- Control colors

Ggraph command

Example

```
GGRAPH
  /GRAPHDATASET
     NAME="graphdataset"
     VARIABLES=age
  /GRAPHSPEC SOURCE=INLINE.
```

What it's for

- /graphdataset creates an intermediate data set (which you never actually see) that is passed to the GPL statements.
- /graphspec specifies where the GPL statements are found

GPL statements

Example:

```
BEGIN GPL
  SOURCE:
      s=userSource(id("graphdataset"))
  DATA: age=col(source(s), name("age"))
  GUIDE: axis(dim(1), label("Age of
      Respondent"))
  GUIDE: axis(dim(2), label("Frequency"))
  ELEMENT:
      interval(position(summary.count(bin.rect(
      age, binWidth(5)))))
  ELEMENT:
      line(position(density.normal(age)),
      color(color.blue))
  ELEMENT:
      line(position(density.kernel.epanechnikov
      (age)), color(color.red))
END GPL.
```

What it's for:

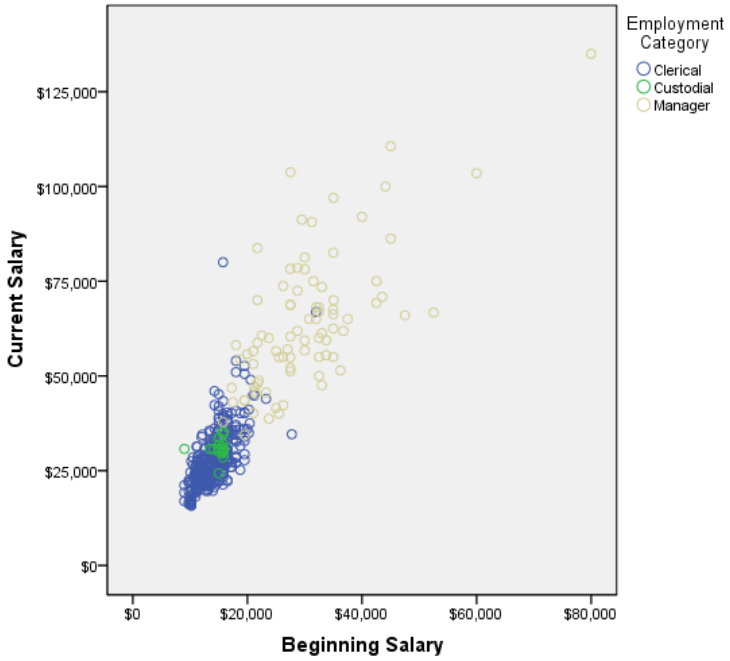
- SOURCE: graph data set
- DATA: variables to use
- GUIDE: axes to draw
- ELEMENT: points, bars, lines, etc.

Ggraph for beginners

- Set up the basic graph with the Chart Builder from the Graphs menu
- Paste the syntax
- Download the documentation and use it to refine your graph
  - Google "IBM SPSS GPL Reference Guide"

Scatter Plots

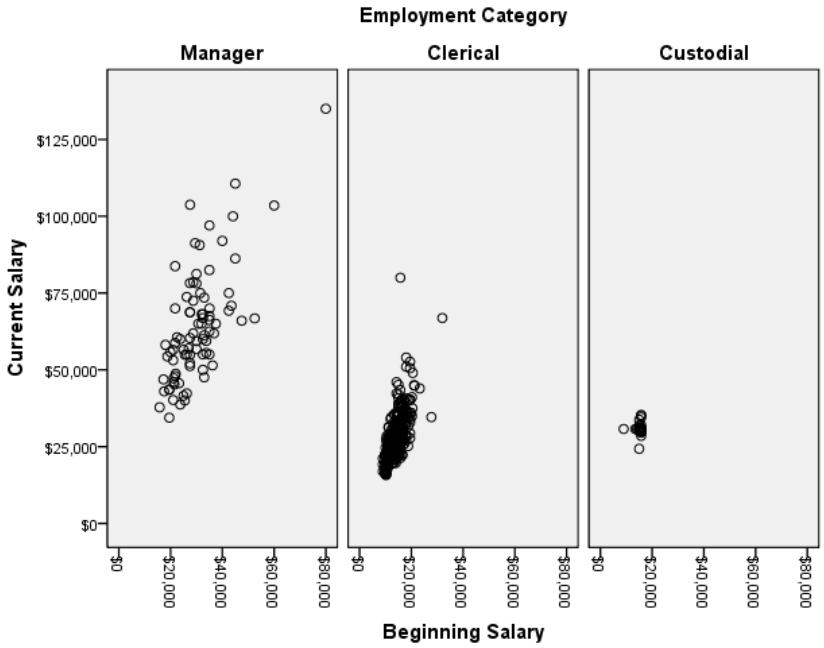
```
get file="y:\spss\data\employee data.sav“.
dataset name ds1.

graph
  /scatterplot=salbegin with salary by
      jobcat.
```



- Note: “x WITH y BY category”

Paneling
(aka “facets” or “trellis”)

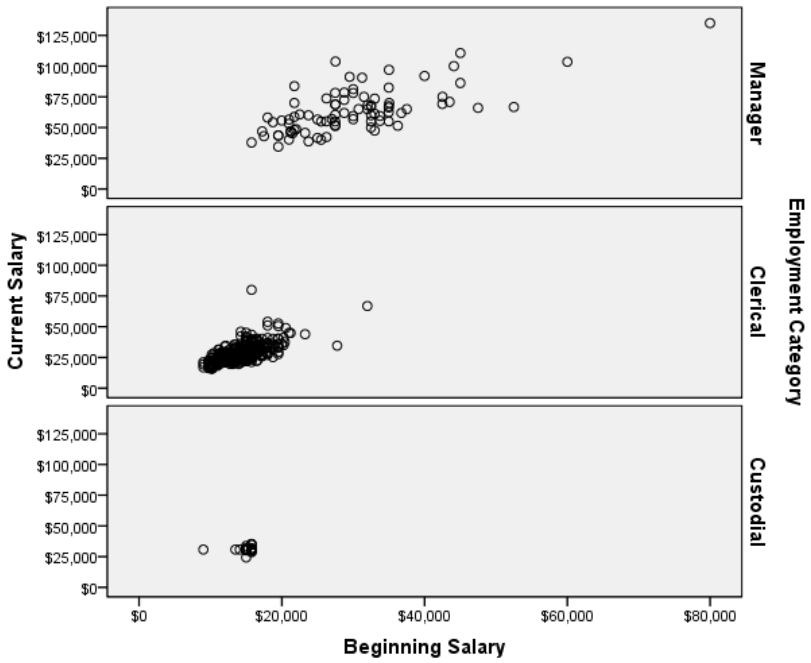
```
graph /*to emphasize Y */
  /scatterplot=salbegin with salary
  /panel colvar=jobcat.
```



You can panel all sorts of graphs, not just scatter plots.

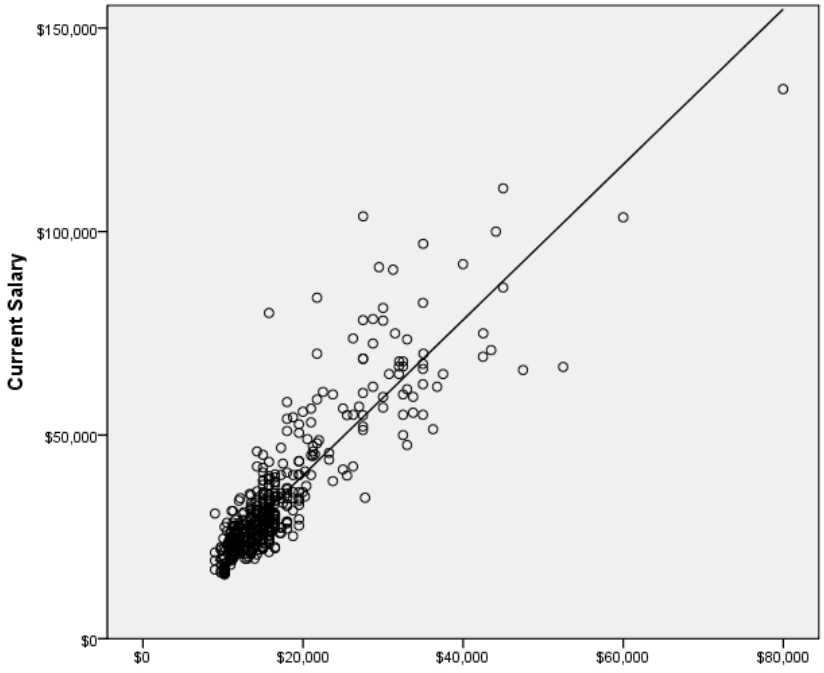
Paneling (cont.)

```
graph /*to emphasize X */
  /scatterplot=salbegin with salary
  /panel rowvar=jobcat.
```



Ggraph scatterplot

```
GGRAPH
  /GRAPHDATASET NAME="graphdataset"
       VARIABLES=salbegin salary
  /GRAPHSPEC SOURCE=INLINE.

BEGIN GPL
  SOURCE: s=userSource(id("graphdataset"))
  DATA: salbegin=col(source(s), name("salbegin"))
  DATA: salary=col(source(s), name("salary"))
  GUIDE: axis(dim(2), label("Current Salary"))
  GUIDE: legend(aesthetic(aesthetic.color.interior),
       label("Employment Category"))
  ELEMENT: point(position(salbegin*salary))
  ELEMENT:
       line(position(smooth.linear(salbegin*salary)))
END GPL.
```



Derived data

- In a basic scatterplot, both x values and y values are given in the data
- In a histogram the y values (counts, percents, densities) must be calculated before they are plotted
- Graphics procedures do ***some*** calculations, but not everything you might want (ggraph does more than graph)
- We often end up deriving data values and saving them as data in order to graph them

Regression example

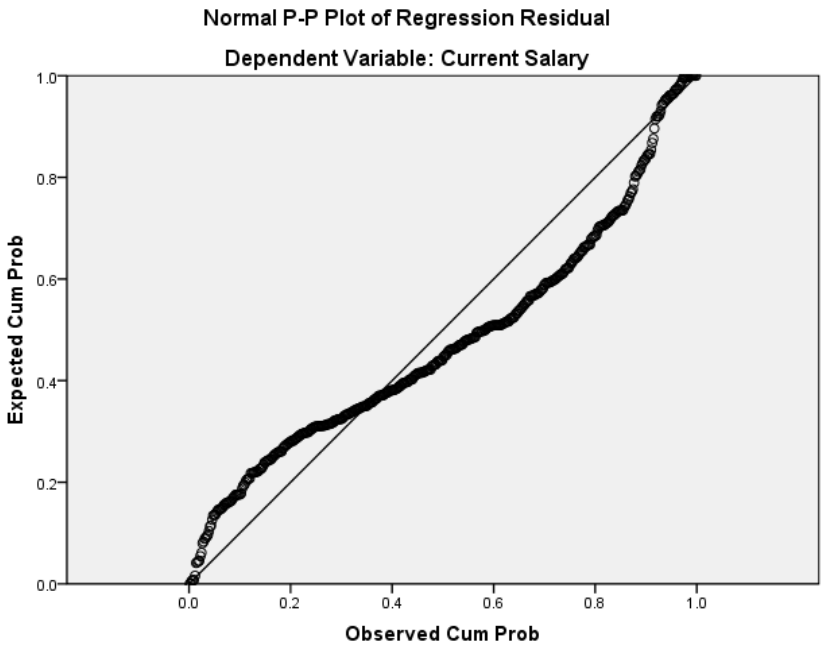
- When judging a regression model, we often want to plot predicted values to visualize our model, and look at the distribution of the residuals to check the model assumptions

```
regression
  /dependent salary
  /method=enter salbegin
  /residuals normprob(resid) /* draws a QQ plot */
  /save pred (predicted) resid (residuals) /*saves derived values*/.
```

Regression QQ plot

- Predicted and residual data values are added to the data set

- Many statistical procedures can optionally produce related graphs

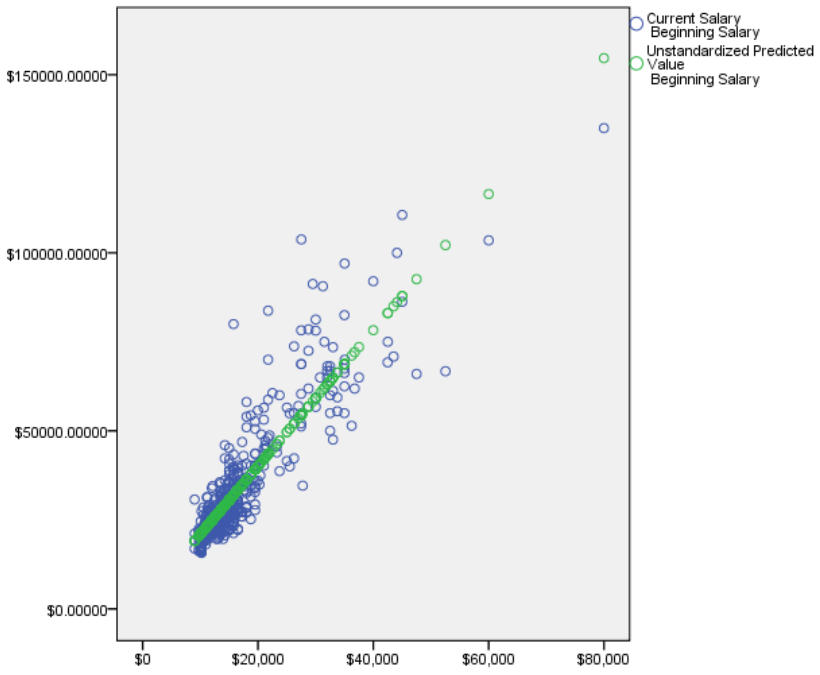


Regression model overlay

graph

/scatterplot(overlay)=salbegin with salary predicted.

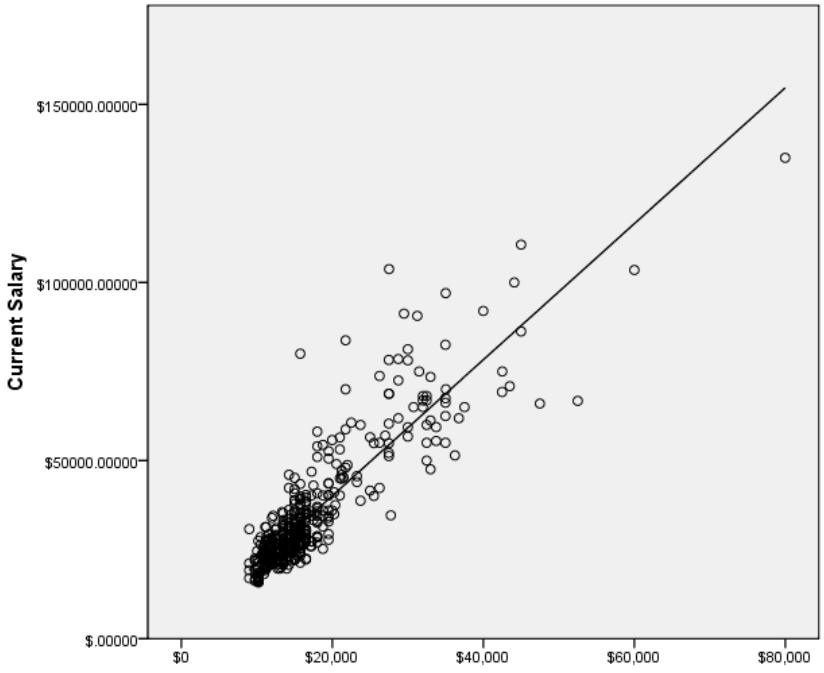
A limitation here is that you overlay points with other points.



Model overlay (cont.)

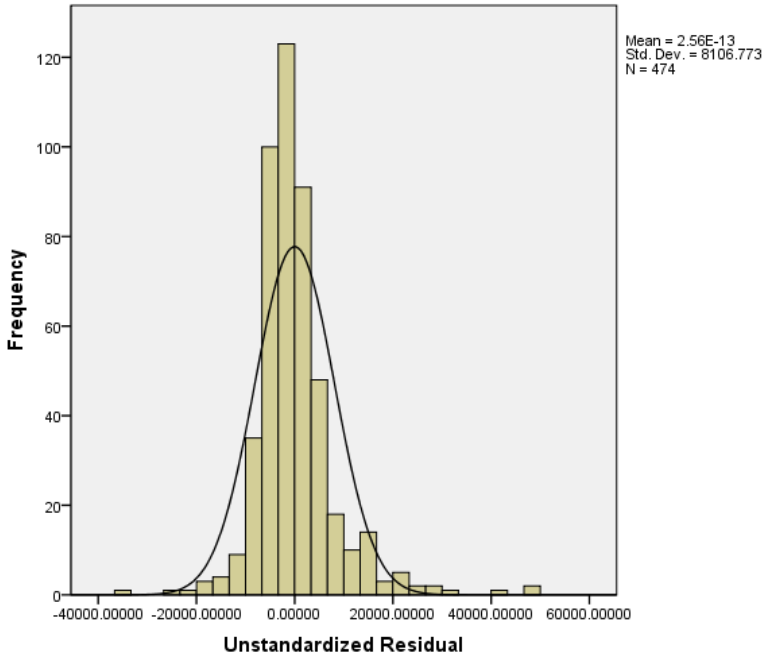
```
GGRAPH
  /GRAPHDATASET NAME="graphdataset"
       VARIABLES=salbegin salary predicted
  /GRAPHSPEC SOURCE=INLINE.
BEGIN GPL
  SOURCE: s=userSource(id("graphdataset"))
  DATA: salbegin=col(source(s), name("salbegin"))
  DATA: salary=col(source(s), name("salary"))
  DATA: predicted=col(source(s), name("predicted"))
  GUIDE: axis(dim(2), label("Current Salary"))
  ELEMENT: point(position(salbegin*salary))
  ELEMENT: line(position(salbegin*predicted))
END GPL.
```

A distinct advantage of GPL is that you can overlay different element types.



Residuals histogram

graph /histogram (normal) = residuals.

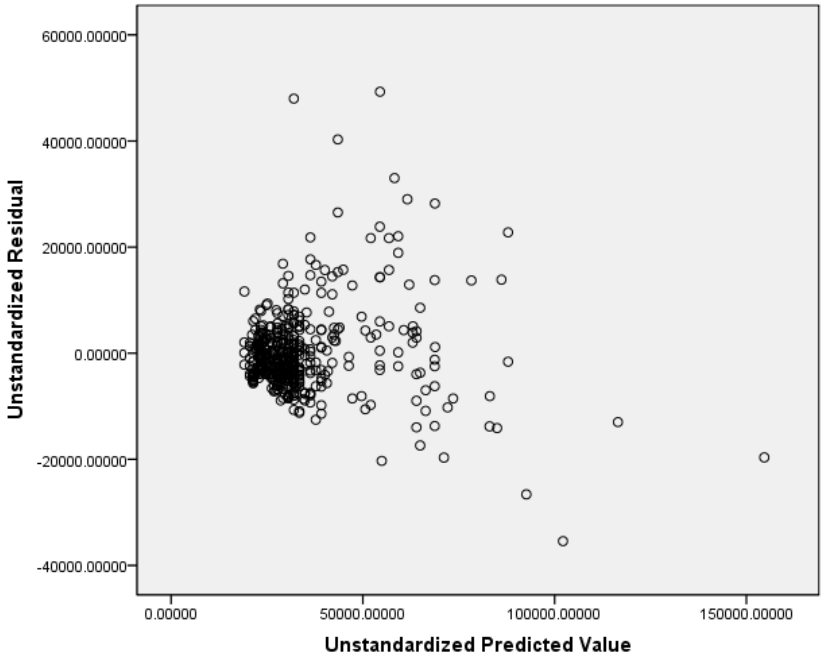


A QQ plot is usually more informative, but the idea behind a histogram is easier for non-statistical audiences to grasp quickly.

Equal variance

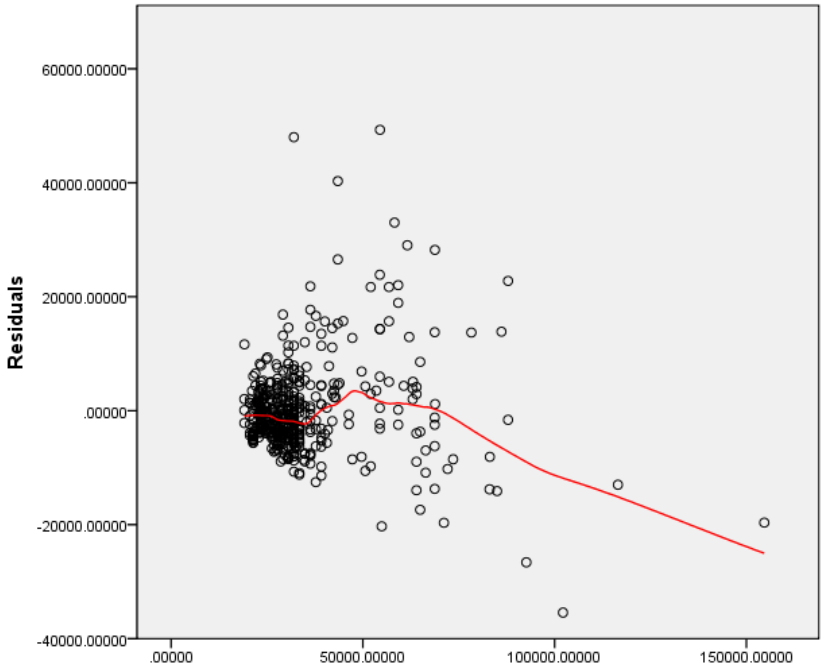
graph /scatterplot = predicted with residuals.

At this point we are working purely with derived data.



Equal variance (cont.)

```
GGRAPH
  /GRAPHDATASET NAME="graphdataset"
       VARIABLES=predicted residuals
  /GRAPHSPEC SOURCE=INLINE.
BEGIN GPL
  SOURCE: s=userSource(id("graphdataset"))
  DATA: predicted=col(source(s), name("predicted"))
  DATA: residuals=col(source(s), name("residuals"))
  GUIDE: axis(dim(2), label("Residuals"))
  ELEMENT: point(position(predicted*residuals))
  ELEMENT:
       line(position(smooth.loess(predicted*residuals)),
       color(color.red))
END GPL.
```



More informative if you can layer on a moving average.

More graph types

- More graph types to explore
  - Bar charts
    - Clustered
    - Stacked
  - Boxplots
  - Scatterplot
    - Matrixes
    - Paneled scatterplots

More graph aesthetics

- Titles
- Axis
  - Labels
  - Numbering
- Element
  - Colors
  - Shapes
- Legends

More data derivation

- Compute new data values
- Save new data values from statistical procedures
- Aggregate data (for summary data sets)
- OMS – any output table can be converted to data